REMARKS

Claims 31-36 are pending in this application. In the Office Action, the Examiner issued a final rejection of all of these claims under 35 U.S.C. §103 as being unpatentable over the prior art, primarily U.S Patent 6,078,744 (Wolczko, et al.). In particular, Claims 31, 33 and 34 were rejected as being unpatentable over Wolczko, et al. in view of U.S. Patent 6,367,072 (Atkinson, et al.), and Claim 32 was rejected as being unpatentable over Wolczko, et al. in vie of U.S. Patent 6,317,872 (Gee, et al.).

The Examiner, in the Office Action, also noted that trademarks should be properly used in the application. In response, Applicants' Attorneys have reviewed the specification, and Applicants herein ask that pages 9, 17 and 22 be amended. On each of these pages, "Java" is being changed to "a Java virtual machine," which is a proper use of the trademark. After reviewing the specification, it is believed that the application properly uses trademarks, and in particular, each use of a trademark is capitalized followed by the appropriate generic language.

In light of the foregoing, the Examiner is asked to reconsider and to withdraw any objection to the specification based on the use therein of trademarks.

With respect to the Claims, Applicants herein ask that independent Claims 31-34 be amended to better define the subject matters of these claims.

For the reasons set forth below, Claims 31-36 patentably distinguish over the prior art and are allowable. The Examiner is, hence, respectfully requested to reconsider and to withdraw the rejections of Claims 31-36 under 35 U.S.C. §103, and to allow these claims.

The present invention relates to a method and system to compile programs or components of a program in a mixed static and dynamic environment. Static compilation involves the process of translating in an off-line manner and generating one or more binary codes to be executed at run-time, and dynamic compilation involves translating a program component to machine code at run-time, before executing that program component. With prior art procedures; there a number of difficulties are encountered when implementing dynamic and static compilation. For example, there are problems associated with performance overhead, dynamic binding and dynamic class loading, testability and serviceability of compilation,

The present invention effectively addresses these problems. In part this is done by providing a two-step process. In a first step, a compiler is used to perform one set of tasks; and then, in a second step, a virtual machine is used to perform further tasks. In that first step, pre-compiled programs are saved, including determining where to place those programs, annotating the programs with dependence information, and processing the programs to produce a further annotated executable code with annotations to help adapt the code to a new executable environment.

Claim 31 is specifically directed to a method for using a virtual machine to execute securely statically compiled code. In this method, a compiler performs a first set of integrity checks, and the virtual machine conducts a second set of integrity checks.

Claim 32 is directed to a procedure for linking precompiled code at run-time within a virtual machine. With this procedure, the compiler maintains certain symbolic entries, and the virtual machine uses these symbolic entries, before the code is executed, to generate direct references in the generated code.

Claim 33 is directed to a method for updating code at run time, when separately compiled code, which contains symbols, changes. With this method, the compiler generates certain code, and the virtual machine may be used to recompile that code under defined circumstances.

Claim 34 defines a method for maintaining compliance with a language requiring dynamic compliance, while still enabling the use of statically generated code for some byte code that depends on byte code that may be separately compiled. In accordance with this method, the compiler performs security features on byte code, and the virtual machine uses those security features to determine if the byte code has changed.

An important feature of the present invention, described in each of Claims 31-34, is the use of precompiled programs, and what is done with those programs.

More specifically, each of Claims 31-34 describes the step of saving pre-compiled programs, including determining where to place those programs, annotating the programs with dependence information, and processing the programs to produce a further annotated executable code with annotations to help adapt the code to a new executable environment.

These claims all differ from Wolczko, et al. in that Wolczko does not disclose the use of precompiled programs, as described in Claims 31-34. These claims describe the use of precompiled programs, not analysis information. The use of precompiled code allows for a better recompilation into native code. As a result, unlike Wolczko, the present invention can dispense with having a compiler in the execution engine.

Applicants' Attorneys have reviewed the other references of record, and these other references, whether considered individually or in combination, also fail to disclose or suggest aspect feature of the invention in the contexts of Claims 31-34.

For instance, Atkinson, et al. was cited for its disclosure of a method and system to help ensure file integrity. This reference teaches incorporating a certification or signature in a file. This certification or signature may be confirmed at the recipient computer.

Gee, et al. discloses a computer architecture for resolving symbolic references in code written in an object oriented programming language.

Atkinson, et al. and Gee, et al, do not disclose or teach using and processing precompiled code as described in Claims 31-34.

This feature of the invention is important because it helps to combine effectively static and dynamic compilation. As a result of this, the invention achieves the reduced performance overhead of dynamic compilers while, at the same time, also achieving the aggressiveness that can be achieved with static compilers.

Because of the above-discussed differences between Claims 31-34 and the prior art, and because of the advantages associated with those differences, Claims 31-34 patentably distinguish over the prior art and are allowable. Claims 35 and 36 are dependent from Claim 31 and are allowable therewith.

The changes to Claims 31-34 that are requested herein only emphasize differences between the claims and the prior art. In particular, these changes describe in more detail the manner in which the pre-compiled programs are used. Further, it is believed that these changes place Claims 31-34 and Claims 35 and 36 in condition for allowance. Consequently, it is believed that entry of this Amendment is appropriate, and such entry is respectfully requested.

The Examiner is, thus, asked to enter this Amendment, to reconsider and to withdraw the rejections of Claims 31-36 under 35 U.S.C. §103, and to allow these claims.

In view of the foregoing, this application is now in condition for allowance, a notice of which is requested. If the Examiner believes that a telephone conference with Applicants' Attorneys would be advantageous to the disposition of this case, the Examiner is asked to telephone the undersigned.

Respectfully Submitted,

John S. Sensny

John S. Sensny
Registration No. 28,757
Attorney for Applicants

Scully, Scott, Murphy & Presser
400 Garden City Plaza
Garden City, New York 11530
(516) 742-4343

JSS:jy